An Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids

Eric J. Nielsen[*], James Lu[†], Michael A. Park[‡], and David L. Darmofal[§]

An algorithm for solving the discrete adjoint system based on an unstructured-grid discretization of the Navier-Stokes equations is presented. The method is constructed such that an adjoint solution exactly dual to a direct differentiation approach is recovered at each time step, yielding a convergence rate which is asymptotically equivalent to that of the primal system. The new approach is implemented within a three-dimensional unstructured-grid framework and results are presented for inviscid, laminar, and turbulent flows. Improvements to the baseline solution algorithm, such as line-implicit relaxation and a tight coupling of the turbulence model, are also presented. By storing nearest-neighbor terms in the residual computation, the dual scheme is computationally efficient, while requiring twice the memory of the flow solution. The scheme is expected to have a broad impact on computational problems related to design optimization as well as error estimation and grid adaptation efforts.

Introduction

The field of computational fluid dynamics (CFD) is increasingly important in the design and validation of new aerodynamic concepts. The use of computational tools can greatly reduce the need for more costly alternatives, such as wind tunnel experiments or flight testing. CFD techniques based on the Navier-Stokes equations have matured into reliable tools for the analysis of complex geometries. However, design optimization using these high-fidelity methods has been greatly inhibited by their relatively high expense and lack of robustness.

In an effort to alleviate the high cost of gradient-based design methodologies, recent work has focused on the use of adjoint methods.[1-20] Adjoint techniques generally fall into one of two categories based on the order in which the discretization and differentiation processes are performed. The two approaches are termed continuous or discrete adjoint methods. For a single output or constraint function, these schemes allow the computation of design sensitivities at the cost of solving a single additional linear problem and a subsequent computation of a matrix-vector product dimensioned by the number of design variables.

In addition to its role in design optimization problems, the solution of the adjoint equation can be used to provide error estimates for an output of engineering interest, as well as grid adaptation information that can be used to improve solution accuracy.[21-29] Traditional methods for grid adaptation have typically relied on ad hoc feature-based quantities.[30-33] In these approaches, features such as shocks, boundary layers, and other regions of interest are characterized by large gradients or curvatures in the solution. The adaptation algorithm then attempts to improve the solution by adding additional grid points in these regions. Unfortunately this approach can yield grids of unwieldy dimensions and even incorrect results.[33] The local flow feature of interest is often over-resolved, while smooth regions of the flowfield are essentially neglected.

The adjoint approach to grid adaptation seeks to minimize the uncertainty or error in some specified output function. In this approach, a local adaptation parameter is obtained by combining flow and adjoint solutions, where the nonlinearities in these solutions are weighted with the local residual error. This adaptation technique implicitly targets the flow features having the highest impact on the output of interest. The test cases examined in Refs. 24, 25, and 27–29 clearly demonstrate the potential of using such an approach to grid adaptation. Comparisons with feature-based techniques are shown and results equivalent to those of uniform grid-refinement studies are obtained at a fraction of the cost. The process terminates when a user-specified error tolerance is achieved.

Unfortunately the solution of the adjoint system of equations for realistic problems has proven to be a formidable task. In Ref. 4, a Krylov method was used to solve the adjoint system for turbulent flows using relatively coarse grids over simple geometries. This work successfully demonstrated the accuracy of the method. However, the computational time required to solve the equations was as much as ten times the cost of the analysis problem, and the scheme failed to converge for many problems. By employing a more extensive preconditioner for the Krylov algorithm, Ref. 5 demonstrated improved performance. However, this preconditioning strategy was shown to require approximately five times the memory of the baseline analysis scheme. This approach proved infeasible on available computers for large-scale problems that require grids containing several million mesh points.

The focus of the current work is to develop a new solution algorithm for the discrete adjoint system described in Refs. 4 and 5. The work is largely based on the recent contributions of Giles,[18–20] in which adjoint solutions for the Euler and Navier-Stokes equations are computed by using an explicit Runge-Kutta scheme combined with multigrid using an exact dual method. Elliott[34] recently used a similar approach for two-dimensional turbulent flows on overset structured grids using multigrid and approximate factorization.

Improvements to the implicit solution technique of Refs. 35 and 36 are described, and an exact dual scheme is developed for the resulting algorithm. Care has been taken to ensure that the implementation yields identical values for linear functionals at each time step, thereby guaranteeing identical asymptotic convergence rates for the primal and dual systems. Results are shown for several small test problems and two large-scale configurations. Convergence of a typical cost function and its derivatives are examined, and computational speed and memory requirements are discussed.

Flow Solution Method

The governing equations are the three-dimensional Reynolds-averaged Navier-Stokes equations. For turbulent flows, the one-equation model of Spalart and Allmaras[38] is used. The flow solver used in the current work is described at length in Refs. 1, 35, and 36. The code uses an implicit, upwind, finite-volume discretization in which the dependent variables are stored at the mesh vertices.

[*]Research Scientist, Computational Modeling and Simulation Branch, NASA Langley Research Center, Hampton, VA 23681, Senior Member AIAA.

[†]Research Assistant, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139, Member AIAA.

[‡]Research Scientist, Computational Modeling and Simulation Branch, NASA Langley Research Center, Hampton, VA 23681, Member AIAA.

[§]Associate Professor, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139, Member AIAA.

Inviscid fluxes at cell interfaces are computed by using the upwind schemes of Roe[39] or Van Leer.[40] Viscous fluxes are formed by using an approach equivalent to a central-difference Galerkin procedure. For steady-state flows, temporal discretization is performed by using a backward Euler time-stepping scheme. A highly scalable parallelization scheme is achieved through domain decomposition and MPI communication.

An approximate solution of the linear system of equations formed at each time step is obtained through several iterations of a point-iterative scheme in which the nodes are updated in an even-odd fashion, resulting in a Gauss-Seidel-type method. This scheme is augmented with a line-relaxation algorithm in the current work.

In Refs. 1–5, 35, and 36, the turbulence model is integrated all the way to the wall without the use of wall functions, and is solved separately from the flow equations at each time step with an identical time-stepping scheme. The resulting linear system is solved with the same iterative scheme employed for the flow equations. The impact of coupling the flow equations and turbulence model will be addressed further in a subsequent section.

In Refs. 1, 4, and 5, a discrete adjoint capability has been developed for the solver. In these references, the discretization of the flow equations and turbulence model described above has been fully differentiated by hand, and the adjoint system of equations has been solved by using a preconditioned GMRES[37] algorithm. The focus of the current work is an alternate solution method based on an exact dual formulation.

Tightly Coupled Turbulence Model

The loosely coupled implementation of the turbulence model in the baseline solver was originally chosen for its convenience in accommodating additional models as they became available. Furthermore, the loose formulation allows for straightforward implicit enforcement of positivity on the eddy viscosity by using M-type matrices and positive operators as described in Ref. 38.

Although the loose formulation has proven satisfactory for engineering-level analysis, it often results in stalled convergence or limit-cycle oscillations that can be detrimental to subsequent adjoint computations. In the current work, a tightly coupled algorithm is used to obtain more robust convergence behavior. The scheme includes the linearizations of the governing flow equations with respect to the turbulence model dependent variable $\tilde{\upsilon}$, as well as the linearizations of the turbulence model with respect to the conserved variables Q, in the computation of the solution updates ΔQ and $\Delta\tilde{\upsilon}$. The approach taken in Ref. 38 to guarantee positivity on $\tilde{\upsilon}$ becomes prohibitively difficult to impose for the tightly coupled system, so that an update clip at $\tilde{\upsilon} = 0$ is necessary to preclude nonphysical behavior as the solution rapidly develops from its initial freestream conditions. This procedure occasionally admits transients in the early stages of a computation which may result in divergence of the solution; these transients are overcome by using small time steps as the initial solution sets up. Although not discussed in the present paper, limited success has been achieved through modifications as suggested in Ref. 41, in which an analysis of the linearized form of the turbulence source terms suggests a similar addition to the diagonal elements of the system.

To demonstrate the effect of coupling on solution convergence, transonic flow over an ONERA M6 wing[42] is computed by using both the loosely and tightly coupled formulations. The grid is shown in Fig. 1 and contains 359,536 nodes and 2,074,955 tetrahedra. The freestream Mach number is 0.84, the angle of attack is $3.06°$, and the Reynolds number is 5 million based on the mean aerodynamic chord. For both cases, the CFL number for the flow and turbulence equations has been linearly ramped from 10 to 200 over the first 50 iterations. The convergence histories for density and turbulence for the two solution methods are shown in Fig. 2. The loosely coupled scheme results in stalled convergence, whereas the tightly coupled scheme steadily reduces the residuals by six orders of magnitude over the course of the computation. Similar results have been observed in other cases, and the tightly coupled algorithm is employed for the remainder of the current work.

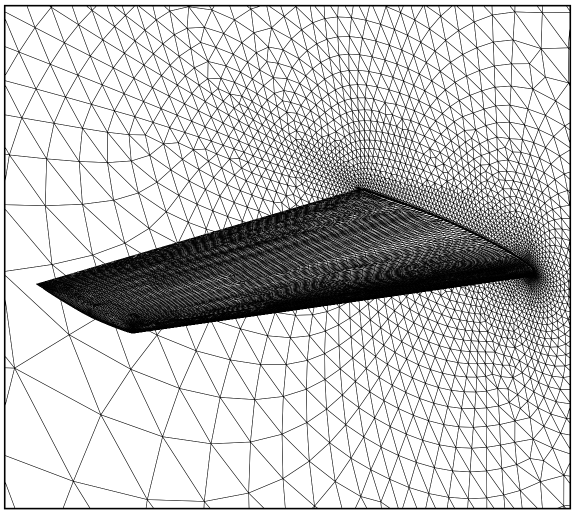
Figure 1. Surface grid for viscous ONERA M6 wing.

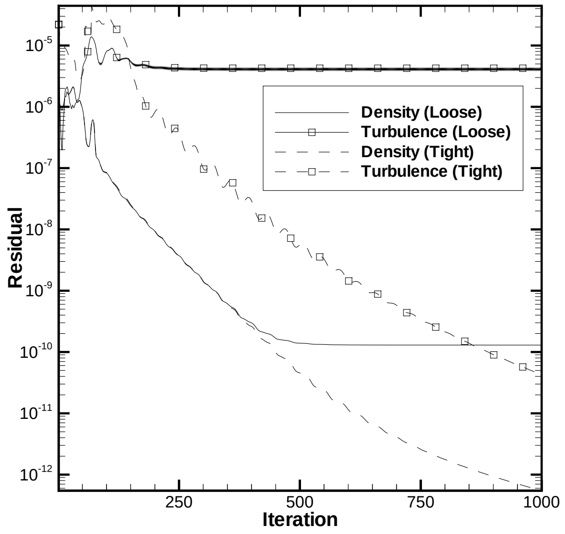

Figure 2. Effect of turbulence model coupling on ONERA M6 flow solution.

Line-Implicit Relaxation Scheme

When used in conjunction with the baseline point-implicit scheme used for the flow solution, a line-implicit scheme can result in improved convergence rates for viscous flows. The benefits of line-relaxation techniques on highly stretched grids are well-known.[43] The central idea is that the coupling of the discrete equations is considerably higher in the direction normal to the grid stretching, so that a relaxation scheme can be constructed to more efficiently solve the equations associated with lines in these directions.

Line Construction Method

On structured grids the line-relaxation direction is typically well-defined because a grid coordinate direction can be readily employed. For unstructured grids, this inherent direction is not available and must be constructed prior to performing any computations. In Ref. 44, implicit lines have been constructed based on neighboring prismatic and hexahedral elements within the boundary layer. In the current work, only tetrahedral elements are employed; thus, an alternative line construction strategy must be used.

To construct a line for implicit relaxation through the boundary layer, an initial point is chosen on a viscous boundary surface. A surface normal is constructed at this point, and the direction of edges connected to this node are compared with the normal using an inner product. The edge with the maximum positive inner product is selected to form the first line segment. The surface normal is

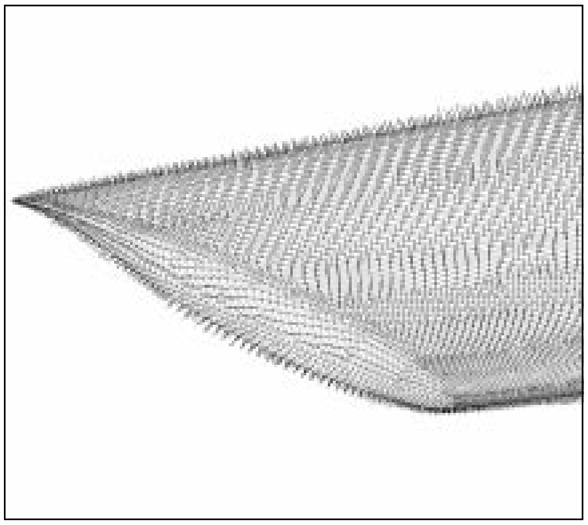
Figure 3. Result of implicit line construction algorithm applied to wingtip geometry of Ref. 45.

then replaced with the direction of the chosen edge, and the process is repeated at the endpoint of successively selected edges, forming the line along which to relax. The process is terminated when the ratio of the length of the longest edge connected to the current node to that of the selected edge falls below a predefined value; a value of 5 is used in the current work. This criterion serves to identify the "inviscid" region of the grid, where elements revert to an isotropic distribution. Construction of a line is also terminated if an edge direction differing by less than 20 degrees from the previous segment cannot be found. This second stopping criterion ensures that line segments remain normal to the original boundary surface. The result of applying this line construction algorithm to a simple wingtip geometry[45] is shown in Fig. 3. A direction suitable for line-implicit relaxation through the boundary layer has been formed at each boundary node, such that approximately 30 grid points are contained in each line.

Another important consideration is the mesh partitioning strategy for parallel processing. To efficiently perform relaxation along any given line, the line should wholly lie within a grid partition. To avoid partition boundaries cutting across relaxation lines, edge weighting is employed in the partitioning phase.[46] This procedure minimizes the number of implicit lines that are split across partition boundaries. In the event that an implicit line is cut by the mesh partitioning, the line is terminated at the partition boundary; no attempt is made to perform line-relaxation across processors. Finally, a multiconstraint version of the partitioning algorithm[47] is used to ensure that load balancing is achieved within the line-implicit and point-implicit regions of the grid.

Line Relaxation Algorithm

Once the linear system of equations at the current time step has been assembled, the unknowns associated with each implicit line are computed exactly by using Gaussian elimination of a local block-tridiagonal system of equations. This procedure is repeated for a series of sweeps over the implicit lines; the initial decomposition of the coefficient matrix is stored so that subsequent sweeps merely consist of a forward/backward substitution procedure. Once a suitable level of convergence is obtained for the unknowns, the remainder of the domain is relaxed by using several sweeps of the baseline point-implicit scheme; adjacent unknowns determined by the line-implicit scheme are taken as known quantities and moved to the right-hand side of the equations.

To demonstrate the benefits of the line-implicit scheme, fully turbulent flow over the wing-body configuration[45] shown in Fig. 4 is computed on twenty-two 2.2 GHz Pentium IV processors using the baseline point-implicit scheme as well as the line-implicit algorithm. The freestream Mach number is 0.75, the angle of attack is $0°$, and the Reynolds number is 3 million based on the mean aerodynamic chord. The grid contains 1,641,452 nodes and 9,650,684

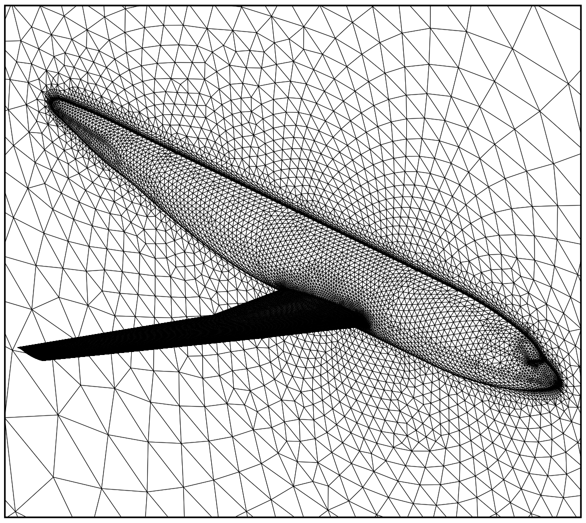
Figure 4. Surface grid for configuration of Ref. 45.

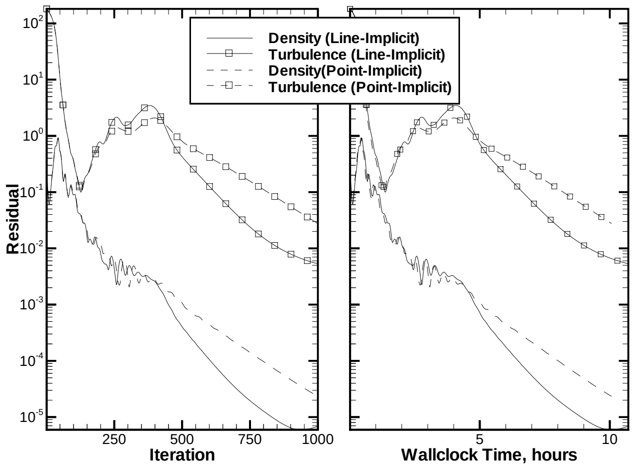

Figure 5. Residual convergence histories for point- and line-implicit solutions.

tetrahedra, and the line construction algorithm places 1,069,238 nodes in the line-implicit region. For this test, 15 sweeps through the line- and point-implicit regions are used for both computations.

Convergence of the density and turbulence residuals for both solution schemes is shown in Fig. 5. The line-implicit strategy ultimately results in a computational savings of approximately 20% over the baseline algorithm. As shown in Fig. 6, lift and drag more rapidly approach their steady-state values with the line-implicit scheme. This behavior is attributed to the rapid development of the boundary layer region, and has been observed in a number of cases. For comparison, the results are plotted in Fig. 7 against experimental values from Ref. 45. Similar to many of the computations reported in Ref. 48, the lift is slightly overpredicted; however, the correlation with the experimental drag polar is good.

Derivation of the Dual Algorithm

Consider the following form of the steady-state nonlinear governing equations, where $\boldsymbol{D}$ and $\boldsymbol{Q}$ represent the vector of design variables and the corresponding dependent variables, respectively, and $\boldsymbol{R}_2$ represents a second-order accurate discretization of the spatial residual.

$$\boldsymbol{R}_2(\boldsymbol{Q}, \boldsymbol{D}) = 0 . \tag{1}$$

Note that in practice, there is also an implicit dependency on the computational grid; the current approach includes these terms, however they are omitted here to simplify the underlying analysis.

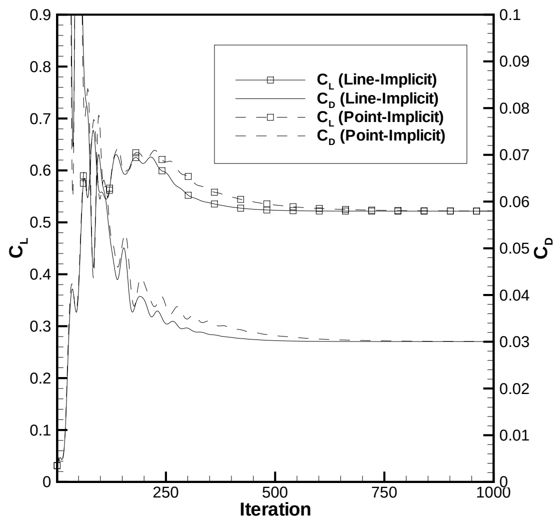

Figure 6. Force convergence for point- and line-implicit solutions.

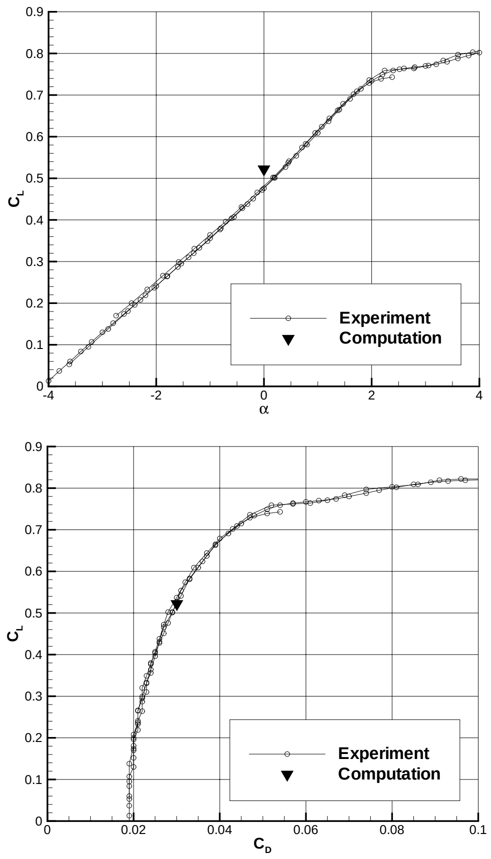

Figure 7. Comparison of computed force coefficients with experimental data of Ref. 45.

An iterative algorithm based on a backward Euler integration scheme for the solution of the nonlinear system of equations given by Eq. 1 can be written as

$$\frac{V}{\Delta t}\boldsymbol{I}(\boldsymbol{Q}^{n+1}-\boldsymbol{Q}^{n})+\boldsymbol{R}_2(\boldsymbol{Q}^{n+1},\boldsymbol{D}) = 0\,, \tag{2}$$

where V and Δt represent the local cell volume and time step, respectively, and $\boldsymbol{Q}^n$ is the vector of dependent variables at time step n. The nonlinear iterative scheme is attained by linearizing the residual about time step n:

$$\left(\frac{V}{\Delta t}\boldsymbol{I}+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}}\right)\Delta Q^n+\boldsymbol{R}_2(\boldsymbol{Q}^n,\boldsymbol{D}) = 0\,, \tag{3}$$

where $\Delta Q^n = Q^{n+1}-Q^n$. Note the iteration superscript n is omitted from the Jacobian matrix $\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}$ for clarity; unless otherwise noted, these derivatives will all be evaluated at time step n.

Note that because the Jacobian used in Eq. 3 is an exact linearization of $\boldsymbol{R}_2$, then for an infinite time step Δt, the iterative scheme corresponds to Newton's method, and $\boldsymbol{Q}^{n+1}$ will converge quadratically to the steady-state value $\boldsymbol{Q}^*$ corresponding to $\boldsymbol{D}$. However, the use of an exact Jacobian $\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}$ is often prohibitively expensive for realistic problems, so that an approximate form of Eq. 3 is typically used:

$$\left(\frac{V}{\Delta t}\boldsymbol{I}+\frac{\partial \boldsymbol{R}_1}{\partial \boldsymbol{Q}}\right)\Delta Q^n+\boldsymbol{R}_2(\boldsymbol{Q}^n,\boldsymbol{D}) = 0\,, \tag{4}$$

where the exact Jacobian has been replaced with a linearization of some first-order approximation to the spatial residual. The drawback to such an approach is that the asymptotic convergence of the algorithm becomes linear in nature.

Equation 4 is a linear system of equations for ΔQ^n, which in principle can be solved exactly. In practice, however, the system is usually solved approximately by using an iterative method. Therefore, Eq. 4 can be restated as

$$\Delta Q^n+\boldsymbol{P}\boldsymbol{R}_2(\boldsymbol{Q}^n,\boldsymbol{D}) = 0\,, \tag{5}$$

where $\boldsymbol{P}$ is an approximation to the inverse of $V/\Delta t\boldsymbol{I}+\partial\boldsymbol{R}_1/\partial\boldsymbol{Q}$, typically achieved through some iterative scheme such as a Gauss-Seidel iteration. In this manner, the asymptotic rate of convergence is governed by the largest eigenvalue of the matrix $\boldsymbol{I}-\boldsymbol{P}\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}$.

Direct Differentiation Algorithm

With minor modifications, the iterative algorithm outlined above can be used to determine the sensitivity derivatives of an output function $f = f(\boldsymbol{Q}^*,\boldsymbol{D})$ with respect to a design variable. Application of the chain rule yields the following:

$$\frac{df}{d\boldsymbol{D}} = \frac{\partial f}{\partial \boldsymbol{Q}^*}\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}+\frac{\partial f}{\partial \boldsymbol{D}}\,. \tag{6}$$

Linearizing Eq. 1 with respect to $\boldsymbol{D}$ gives the linear residual equations for the sensitivity derivatives $\partial\boldsymbol{Q}^*/\partial\boldsymbol{D}$:

$$\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} = 0\,. \tag{7}$$

Applying the algorithm of Eq. 3 to this expression gives an iterative scheme for $\partial\boldsymbol{Q}^*/\partial\boldsymbol{D}$:

$$\left(\frac{V}{\Delta t}\boldsymbol{I}+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}}\right)\Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} = 0\,. \tag{8}$$

Finally, by replacing the exact Jacobian with the approximate Jacobian as before and performing an approximate inverse gives the final form of the direct differentiation iterative algorithm for the sensitivity derivatives $\partial\boldsymbol{Q}^*/\partial\boldsymbol{D}$:

$$\Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n+\boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n+\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right] = 0\,. \tag{9}$$

Note the asymptotic rate of convergence is again dictated by the largest eigenvalue of $\boldsymbol{I}-\boldsymbol{P}\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}$, and therefore will be equivalent to that of the scheme used to attain $\boldsymbol{Q}^*$. After N iterations of Eq. 9, the derivative of the output f with respect to the design variables $\boldsymbol{D}$ may be approximated as

$$\frac{df}{d\boldsymbol{D}} = \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N+\frac{\partial f}{\partial \boldsymbol{D}}\,. \tag{10}$$

Exact Dual Algorithm

In this section, an iterative solution of the dual problem is constructed in a manner which guarantees the functional estimate is identical at every iteration to that obtained from the direct differentiation algorithm. Using an adjoint approach, the sensitivity derivative of the output can be written as

$$\frac{df}{d\boldsymbol{D}} = -\Lambda^T \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} + \frac{\partial f}{\partial \boldsymbol{D}}, \tag{11}$$

where Λ is an adjoint variable that satisfies

$$\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right]^T \Lambda - \left[\frac{\partial f}{\partial \boldsymbol{Q}^*}\right]^T = 0 . \tag{12}$$

Since Eq. 12 is independent of the vector $\boldsymbol{D}$, the adjoint approach is particularly attractive for aerodynamic design problems with a large number of design variables and relatively few constraints. By examining Eqs. 7 and 12, it can be seen that

$$\frac{\partial f}{\partial \boldsymbol{Q}^*}\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}} = -\Lambda^T \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} . \tag{13}$$

Therefore, the values of $df/d\boldsymbol{D}$ calculated from Eqs. 6 and 11 are identical. However, since an iterative algorithm is used to estimate $\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}$, this relationship will not hold in general. Following Giles' exact dual approach, an iterative adjoint solution algorithm is sought which satisfies

$$\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = -(\Lambda^N)^T \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} , \tag{14}$$

where N is some iteration at which the equality is to be enforced. However, the resulting exact dual algorithm will not depend on N, and the expression given by Eq. 14 will be valid at every iteration. In this sense, the adjoint iterative algorithm will be exactly dual to the direct differentiation algorithm.

Introducing Lagrange multipliers, the left-hand side of Eq. 14 after N iterations can be written as

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N &= \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N \\ &\quad - \sum_{n=0}^{N-1} (\lambda^{n+1})^T \left\{ \Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right]\right\}\end{aligned} \tag{15}$$

Then, by defining

$$\Lambda^n = \sum_{m=N-n}^{N-1} \boldsymbol{P}^T \lambda^{m+1} , \tag{16}$$

and performing the manipulations shown in the appendix, the exact duality condition requires that

$$\Lambda^{n+1} - \Lambda^n + \boldsymbol{P}^T\left[\left(\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right)^T \Lambda^n - \left(\frac{\partial f}{\partial \boldsymbol{Q}^*}\right)^T\right] = 0 \tag{17}$$

with an initial condition $\Lambda^0 = 0$. As mentioned previously, Eq. 17 is independent of N and thus the exact duality condition holds for all values of N. This condition guarantees an asymptotic rate of convergence for the dual problem which is governed by the largest eigenvalue of $\boldsymbol{I} - \boldsymbol{P}^T[\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}^*]^T$ and is therefore ultimately equivalent to that of the solution for $\boldsymbol{Q}^*$.

Relationship Between ***P*** *and* $\boldsymbol{P}^T$

In the current work, the approximate inverse $\boldsymbol{P}$ is the result of applying multiple iterations of a fixed-point method, which can be written in the following manner:

$$\boldsymbol{M}\boldsymbol{x}^{j+1} = \boldsymbol{b} + \boldsymbol{N}\boldsymbol{x}^j \tag{18}$$

or

$$\boldsymbol{x}^{j+1} - \boldsymbol{x}^j = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^j) , \tag{19}$$

where $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$. Here, $\boldsymbol{M}$ is some matrix that is easily invertible and approximates $\boldsymbol{A}$.

Given some initial estimate $\boldsymbol{x}^0$, after J iterations the scheme results in the following:

$$\boldsymbol{x}^J - \boldsymbol{x}^0 = [\boldsymbol{I} - (\boldsymbol{M}^{-1}\boldsymbol{N})^J \boldsymbol{A}^{-1}](\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0) . \tag{20}$$

Therefore, with respect to Eq. 9, the approximate inverse $\boldsymbol{P}$ using J iterations of the fixed-point method is

$$\boldsymbol{P} = \boldsymbol{I} - (\boldsymbol{M}^{-1}\boldsymbol{N})^J \boldsymbol{A}^{-1} . \tag{21}$$

Forming the transpose of $\boldsymbol{P}$ yields

$$\boldsymbol{P}^T = \boldsymbol{I} - \boldsymbol{A}^{-T}[(\boldsymbol{M}^{-1}\boldsymbol{N})^T]^J . \tag{22}$$

Becase $\boldsymbol{M}^{-1}\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$, then

$$\boldsymbol{P}^T = \boldsymbol{I} - \boldsymbol{A}^{-T}[(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^T]^J ,$$

or

$$\boldsymbol{P}^T = \boldsymbol{I} - [\boldsymbol{M}^{-T}\boldsymbol{N}^T]^J \boldsymbol{A}^{-T} . \tag{23}$$

If the dual problem takes the form $\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{g}$ as in Eq. 17, then analogous to Eq. 20, Eq. 23 implies that the fixed-point iterative scheme for the dual problem is

$$\boldsymbol{y}^{j+1} - \boldsymbol{y}^j = \boldsymbol{M}^{-T}(\boldsymbol{g} - \boldsymbol{A}^T\boldsymbol{y}^j) . \tag{24}$$

For a point-Jacobi scheme, the transpose operator applied to the iteration matrix $\boldsymbol{M}$ is of no consequence, other than to imply that the diagonal blocks are to be transposed. However, for a Gauss-Seidel iterative method such as the one used in the current work, the transpose operation transforms an upper-triangular iteration matrix into a lower-triangular form, and vice versa. Therefore, the transpose implies that even-odd sweeps through the system of equations must be performed in a reverse manner. The same argument holds for the line-implicit operator used to relax the boundary layer region, although the solution within each line can be obtained in the same manner as the baseline scheme because the inversion is exact for the set of local equations within the line.

Implementation of the Dual Algorithm

The majority of the effort associated with developing an adjoint solver is in forming the exact linearizations of the second-order residual. For the codes used in the current study, this differentiation has been previously performed and the accuracy has been established in Refs. 1, 3, and 5. The implementation of the exact dual algorithm is primarily a high-level task that involves the manipulation of existing routines; this process is made easier through the programming practices established in Ref. 49.

After the flow solution has completed a series of n time steps using the algorithm given by Eq. 5, the adjoint solve can be performed in an analogous fashion according to Eq. 17. Note that the dual iterative scheme is essentially identical to the baseline analysis scheme so that little extra coding is required.

According to Eq. 17, the exact linearization of the second-order spatial residual is required for the adjoint computation. However, since these terms only appear in a matrix-vector product, they can be formed on a term-by-term basis and therefore do not require the extensive storage typically associated with the higher-order stencil. The transpose of the first-order Jacobian matrix $(V/\Delta t)\boldsymbol{I} + \partial \boldsymbol{R}_1/\partial \boldsymbol{Q}^*$ is used to solve the system of adjoint equations, so that the storage required by the exact dual scheme is roughly equivalent to that of the baseline analysis code. This requirement is in contrast to the solution algorithm outlined in Ref. 5, in which the complete linearization of the second-order residual $[\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}^*]^T$ was used as a preconditioner. The terms arising from this larger stencil resulted in a memory requirement approximately

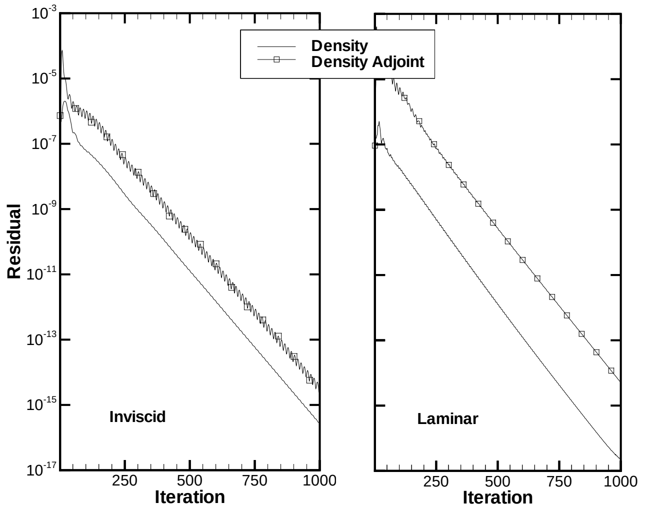

Figure 8. Density residual histories for inviscid and laminar flow over ONERA M6 wing.

five times that of the scheme outlined in Refs. 35 and 36. Finally, since the Jacobian matrix $\partial \boldsymbol{R}_1 / \partial \boldsymbol{Q}^*$ is constant for an adjoint computation, the block and tridiagonal-block decompositions used for the point- and line-implicit iterations need only be performed once and stored.

The solution strategy outlined in Refs. 1 and 4 required a frequent computation of the form $[\partial \boldsymbol{R}_2 / \partial \boldsymbol{Q}^*]^T \Delta\Lambda$ for the Krylov method being used. The viscous terms arising from the nearest-neighbor discretization were stored, so that only the inviscid terms involving the larger stencil were recomputed for each new Krylov vector. In the current work, an analogous approach can be used in the formation of the adjoint residual component $[\partial \boldsymbol{R}_2 / \partial \boldsymbol{Q}^*]^T \Lambda^n$ if additional memory is available. Furthermore, if sufficient memory is available for the higher-order terms as used for the preconditioner in Ref. 5, the complete linearization may be stored. In this manner, the residual update at time step n reduces to an explicit matrix-vector product. Since all terms comprising the residual are stored in this approach, the solution time for the adjoint problem could be reduced considerably. Solution times and memory requirements for each of these strategies will be shown in a subsequent section.

Validation Cases

A series of small test cases involving inviscid, laminar, and turbulent flow over the ONERA M6 wing geometry is presented to verify that the exact dual algorithm has been implemented correctly. Each of the viscous tests has been run using the line-implicit relaxation scheme. For each example, the property of Eq. 14 has been verified to machine accuracy, and the asymptotic convergence rates for density (and turbulence where appropriate) are shown to be equal to that of the corresponding adjoint equations. For visual clarity, the convergence histories of the momentum and energy equations and their adjoint counterparts are not shown in the figures that follow, but have been found to closely follow that of the density equation. The cost function for each test case is based on lift, and the CFL number for the flow solution has been linearly ramped from 10 to 200 over the first 50 iterations. Unless otherwise stated, each of the computations shown here has been performed on sixteen 2.2 GHz Pentium IV processors.

Inviscid Wing

The first test case is inviscid flow, with a freestream Mach number equal to 0.84 and an angle of attack of $3.06°$. The grid for this test contains 357,900 nodes and 2,000,034 tetrahedra. The convergence histories of the density equation for the flow and adjoint solutions are shown in Fig. 8; the convergence rates are equivalent.

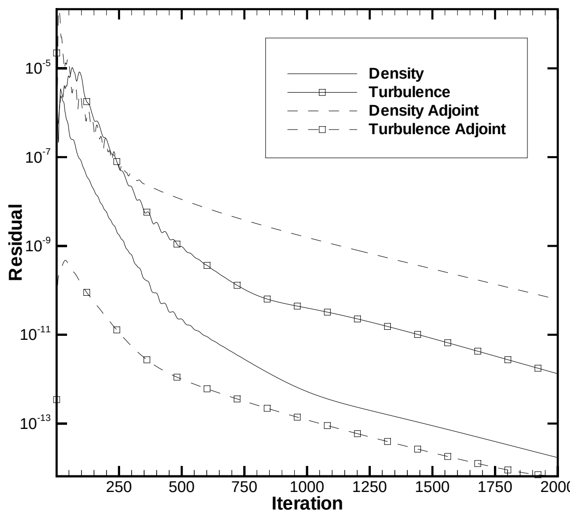

Figure 9. Density and turbulence residual histories for turbulent flow over ONERA M6 wing.

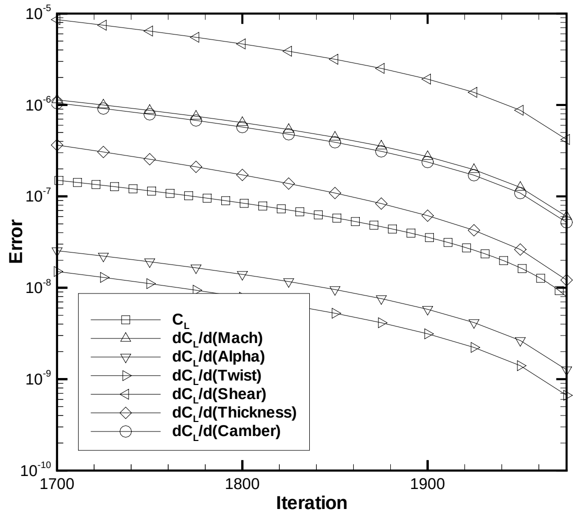

Figure 10. Convergence of lift and lift derivatives for turbulent flow over ONERA M6 wing.

Laminar Wing

The first viscous case is for laminar flow with a freestream Mach number of 0.3, an angle of attack of $2°$, and a Reynolds number of 1000 based on the mean aerodynamic chord. The grid shown in Fig. 1 is used for this example. The density convergence histories for the flow and adjoint solutions are shown in Fig. 8, and the convergence rates are identical.

Turbulent Wing

The final validation case is for turbulent flow over the ONERA M6 wing on the same grid that was used in the previous example. For this case, the freestream Mach number is 0.84, the angle of attack is $3.06°$, and the Reynolds number is 5 million based on the mean aerodynamic chord. The convergence histories for the flow and adjoint equations shown in Fig. 9 exhibit similar asymptotic rates.

To further demonstrate the asymptotic nature of the flow and adjoint solution algorithms, the turbulent flow test case is also used to examine the convergence of a cost function and its derivatives. For this test, the wing has been parameterized by using the method of Ref. 50. Fig. 10 shows the error in the lift coefficient as the flow solution converges; it also shows the error in the derivatives of lift with respect to freestream Mach number, angle of attack, and several shape design variables at the midspan location as the adjoint solution converges. For this case, the error is defined as the differ-

Table 1. Memory and CPU requirements for flow solver and various linearization storage strategies used for computing adjoint residual.

Solver	Memory (GB)	Wallclock Time (Hours)
Flow Solver	1.5	50.4
Adjoint Solver:		
Explicit computation of all residual terms	1.6	42.0
Explicit computation of inviscid residual terms; viscous, turbulent residual terms stored	3.1	25.9
All residual terms stored	10.8	17.1

ence between the current value and the final converged result. The errors are reduced at a similar rate for each computation.

The turbulent flow test case is also used to evaluate the linearization storage strategies described above. Tests are performed on sixteen 250 MHz R10000 processors of a Silicon Graphics Origin 2000 system to simplify memory monitoring and to eliminate the effects of network traffic. The turbulent flow validation case has been repeated using each of the storage strategies; memory and wallclock statistics are shown in Table 1.

Recomputing all of the necessary linearizations for the adjoint residual yields a memory requirement roughly equivalent to that of the flow solution, while the wallclock time is approximately 17% less. The discrepancy in solution times can be attributed to several factors. Forming the flux Jacobians for the flow solution is roughly equivalent in cost to a residual evaluation for the adjoint equations, with a slight penalty for the computation of the second-order inviscid terms required by the latter. However, in the current implementation these second-order terms merely require an application of the chain rule during computation of the first-order inviscid linearizations. Note that the flux Jacobians used for the left-hand side are fixed during an adjoint solution, so that the initial cost of forming these terms and performing the block-decompositions and block-tridiagonal decompositions a priori is amortized over the duration of the computation. In addition to the flux Jacobians required at each time step of a flow solution, a residual evaluation is necessary, which requires updated solution gradients and other solution-dependent terms such as edge weights. Finally, the flow solution also incurs a small amount of overhead at each time step in the computation of current forces and moments.

Alternatively, by storing the nearest-neighbor adjoint residual contributions that arise from the linearization of the viscous and turbulent terms, the wallclock time for an adjoint solution can be reduced to approximately half that of the flow solution while doubling the memory requirement. Finally, by storing the complete linearization of the second-order residual, the computational time is reduced to roughly one-third of the baseline flow solution; however, the memory requirement for this approach is a factor of 6 higher than the baseline flow solution algorithm. Based on these results, the viscous and turbulent terms are stored throughout the remainder of the current work, and the inviscid contributions are recomputed as needed.

Large-Scale Test Cases

Two large-scale test cases are used to evaluate the solution algorithm on realistic configurations. Each of the grids has been generated by using the method described in Ref. 51. In the interest of comparing asymptotic convergence behavior, solutions have been converged considerably beyond the usual requirements for engineering-level answers; the stated run times do not represent time required to obtain a sufficiently accurate solution.

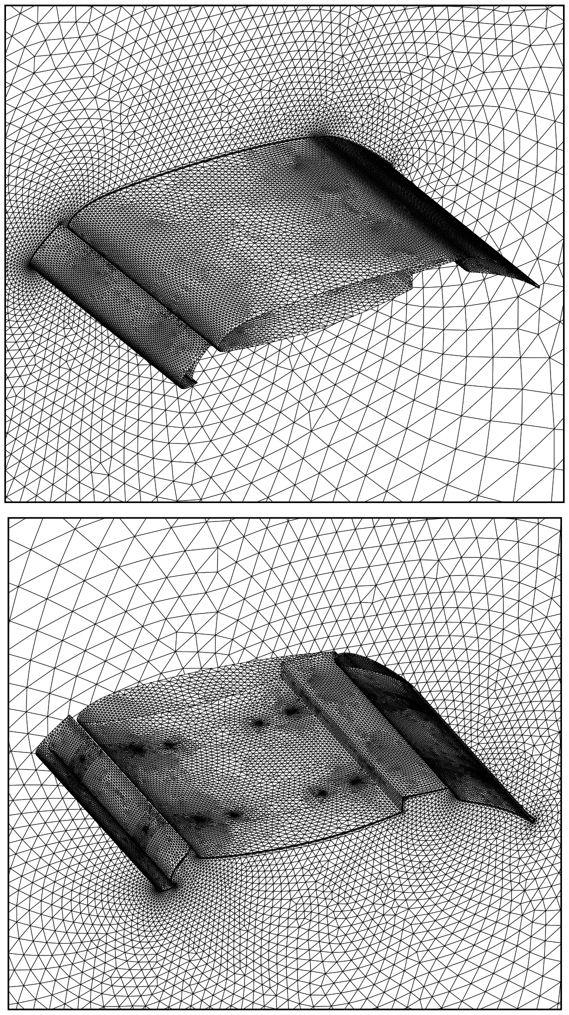

Figure 11. Surface grid for high-lift configuration.

High-Lift Configuration

Turbulent flow over the high-lift configuration[52,53] shown in Fig. 11 is computed using eighteen 1.7 GHz Pentium IV processors. The grid contains 846,863 nodes and 4,879,086 cells, and the implicit line construction algorithm places 418,523 nodes in the line-implicit region. Although only a single plane is shown, symmetry plane boundary conditions are used on both sides of the configuration. The freestream Mach number is 0.2, the angle of attack is $12°$, and the Reynolds number is 9 million based on the stowed chord length. The CFL numbers for the flow equations and turbulence model are linearly ramped from 10 to 200 and 1 to 20, respectively, over the first 200 iterations. For this case, the objective function is based on the lift coefficient. The flow solver requires approximately 3.5 GB of memory and 36.2 hours of wallclock time for the current example; the adjoint solver requires roughly 7.1 GB and 19.2 hours. The convergence histories for the flow and adjoint solutions are shown in Fig. 12; the asymptotic rates are similar.

Modern Transport Configuration

For this test, transonic flow over the modern transport configuration shown in Fig. 13 is considered. The grid for this case contains 1,731,262 nodes and 10,197,838 cells. The implicit line construction algorithm places 1,220,567 nodes in the line relaxation region. The freestream Mach number for this case is 0.84, the angle of attack is $2.25°$, and the Reynolds number is 3 million based on the mean aerodynamic chord. The computations shown here are performed on twenty-three 2.4 GHz Pentium IV processors, and the cost function is based on the drag coefficient.

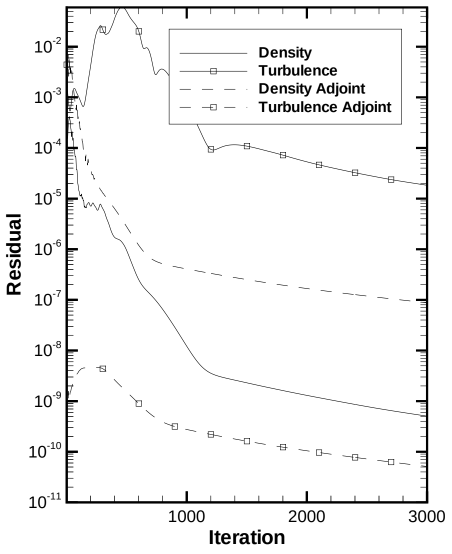

Figure 12. Density and turbulence residual histories for turbulent flow over high-lift configuration.

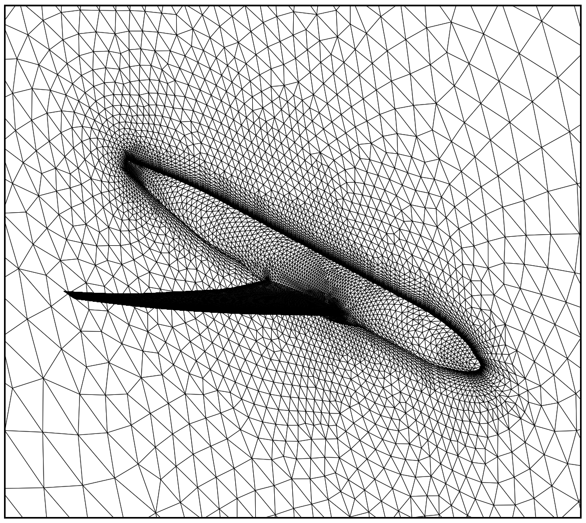

Figure 13. Surface grid for modern transport configuration.

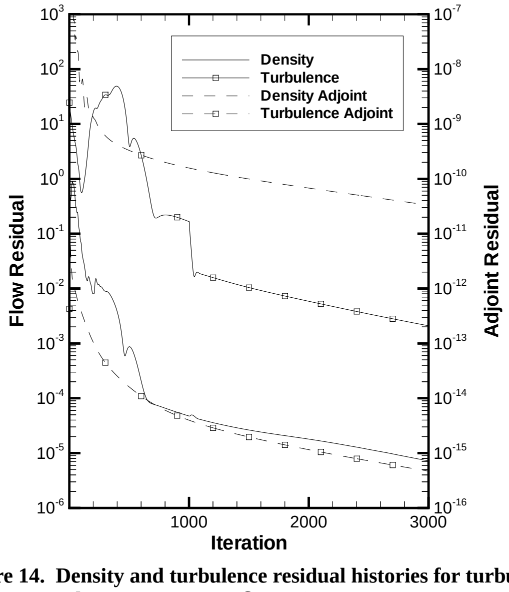

Figure 14. Density and turbulence residual histories for turbulent flow over modern transport configuration.

The convergence histories for the flow and adjoint solutions are shown in Fig. 14. The CFL numbers for the flow equations and turbulence model are linearly ramped from 10 to 200 and 1 to 20, respectively, over the first 200 iterations. The CFL number used for the turbulence model is increased to 200 after the first 1000 iterations. As shown in Fig. 14, the asymptotic rates for both computations are similar. For this example, the flow solution requires 7.4 GB of memory and 41.8 hours of wallclock time; the adjoint solution requires 15.0 GB of storage and 32 wallclock hours.

Summary and Future Work

An adjoint solution algorithm that preserves discrete duality for turbulent flows on unstructured grids has been developed and implemented. The scheme is based on a backward-Euler discretization of the Reynolds-averaged Navier-Stokes equations with a tightly coupled one-equation turbulence model, where the linear problem at each time step is solved with a line-implicit relaxation in the boundary layer and a point-implicit technique through the remainder of the domain.

Results for several cases show that the exact dual scheme achieves asymptotic convergence behavior equivalent to that of the flow problem. By storing the viscous and turbulent contributions to the adjoint residual, the memory required for the algorithm is approximately twice that of the flow solution, but with execution times roughly 25–50% less for an equivalent number of iterations.

Efforts are currently underway to develop a multigrid implementation for improved convergence rates and to extend the discretization to include mixed-element grids. Real gas physical models from existing hypersonic structured-grid solvers are also being added, with the long-term goal of achieving a self-adaptive analysis and design capability valid across the speed range.[54]

Acknowledgements

The authors wish to thank Drs. James Thomas, Kyle Anderson, Steve Allmaras, and Peter Gnoffo for many helpful discussions pertaining to the current work. Dr. Jamshid Samareh and Elizabeth Lee-Rausch are gratefully acknowledged for providing the geometric parameterization and high-lift grid, respectively. The study would not have been possible without the computational resources provided by Sally Viken and Bil Kleb, and Drs. Mark Carpenter, Veer Vatsa, and Dick Wilmoth.

Appendix: Derivation of Exact Dual Outer Iteration

In this appendix, the expression given by Eq. 15 is manipulated to give the exact dual iterative scheme as shown in Eq. 17. The derivation shown here is identical to that of Ref. 18.

Using the discrete form of integration by parts, namely

$$\sum_{n=0}^{N-1} a^{n+1}(b^{n+1}-b^n) = a^N b^N - a^0 b^0 - \sum_{n=0}^{N-1}(a^{n+1}-a^n)b^n\,, \quad (25)$$

Eq. 15 can be expanded as

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = {} & \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N - (\lambda^N)^T\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N + (\lambda^0)^T\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^0 \\ & + \sum_{n=0}^{N-1}\left\{(\lambda^{n+1}-\lambda^n)^T\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n \right.\\ & \left. + (\lambda^{n+1})^T \boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right]\right\}\end{aligned} \quad (26)$$

Rearranging terms and applying the initial condition $[\partial \boldsymbol{Q}^*/\partial D]^0 = 0$ gives

$$\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = \left[\frac{\partial f}{\partial \boldsymbol{Q}^*} - (\lambda^N)^T\right]\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N + \sum_{n=0}^{N-1}\left\{\left[(\lambda^{n+1} - \lambda^n)^T + \left(\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right)^T \boldsymbol{P}^T \lambda^{n+1}\right]\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n - \boldsymbol{P}^T \lambda^{n+1}\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right\} \quad (27)$$

By applying the condition $\lambda^N = (\partial f / \partial \boldsymbol{Q}^*)^T$ and the variable substitution shown in Eq. 16, the exact dual iteration takes the form of Eq. 17 through the elimination of the terms involving $\partial \boldsymbol{Q}^* / \partial D$.

References

[1]Nielsen, E.J., "Aerodynamic Design Sensitivities on an Unstructured Mesh Using the Navier-Stokes Equations and a Discrete Adjoint Formulation," Ph.D. Dissertation, Dept. of Aerospace and Ocean Engineering, Virginia Polytechnic Inst. and State Univ., December 1998.

[2]Anderson, W.K., and Bonhaus, D.L., "Airfoil Design on Unstructured Grids for Turbulent Flows," *AIAA Journal*, Vol. 37, No. 2, 1999, pp. 185–191.

[3]Anderson, W.K., and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers and Fluids*, Vol. 28, No. 4, 1999, pp. 443–480.

[4]Nielsen, E.J., and Anderson, W.K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, 1999, pp. 1411–1419.

[5]Nielsen, E.J., and Anderson, W.K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[6]Jameson, A., Pierce, N.A., and Martinelli, L., "Optimum Aerodynamic Design Using the Navier-Stokes Equations," AIAA Paper 97-0101, January 1997.

[7]Reuther, J.J., Jameson, A., Alonso, J.J., Rimlinger, M.J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60.

[8]Elliott, J., and Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes," *AIAA Journal*, Vol. 35, No. 9, 1997, pp. 1479–1485.

[9]Soemarwoto, B., "Multipoint Aerodynamic Design by Optimization," Ph.D. Dissertation, Dept. of Theoretical Aerodynamics, Delft University of Technology, December 1996.

[10]Mohammadi, B., "Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence," AIAA Paper 97-0099, January 1997.

[11]Nemec, M., and Zingg, D.W., "Towards Efficient Aerodynamic Shape Optimization Based on the Navier-Stokes Equations," AIAA Paper 2001-2532, 2001.

[12]Kim, H.-J., Sasaki, D., Obayashi, S., and Nakahashi, K., "Aerodynamic Optimization of Supersonic Transport Wing Using Unstructured Adjoint Method," *AIAA Journal*, Vol. 39, No. 6, June 2001, pp. 1011–1020.

[13]Soto, O., and Lohner, R., "A Mixed Adjoint Formulation for Incompressible Turbulent Problems," AIAA Paper 2002-0451, 2002.

[14]Sung, C., and Kwon, J.H., "Aerodynamic Design Optimization Using the Navier-Stokes and Adjoint Equations," AIAA Paper 2001-0266, 2001.

[15]Kim, C.S., Kim, C., and Rho, O.H., "Sensitivity Analysis for the Navier-Stokes Equations with Two-Equation Turbulence Models," *AIAA Journal*, Vol. 39, No. 5, May 2001, pp. 838–845.

[16]Iollo, A., Salas, M.D., and Ta'asan, S., "Shape Optimization Governed by the Euler Equations Using an Adjoint Method," ICASE Report No. 93-78, November 1993.

[17]Newman III, J.C., Taylor III, A.C., and Burgreen, G.W., "An Unstructured Grid Approach to Sensitivity Analysis and Shape Optimization Using the Euler Equations," AIAA Paper 95-1646, 1995.

[18]Giles, M.B., "On the Use of Runge-Kutta Time-Marching and Multigrid for the Solution of Steady Adjoint Equations," AD2000 Conference in Nice, June 19–23, 2000.

[19]Giles, M.B., "Adjoint Code Developments Using the Exact Discrete Approach," AIAA Paper 2001-2596, 2001.

[20]Giles, M.B., "On the Iterative Solution of Adjoint Equations," in *Automatic Differentiation: From Simulation to Optimization*, G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, eds., Springer-Verlag, 2001.

[21]Monk, P., and Suli, E., "The Adaptive Computation of Far-Field Patterns by A Posteriori Error Estimation of Linear Functionals," *SIAM Journal on Numerical Analysis*, Vol. 8, 1998, pp. 251–274.

[22]Paraschivoiu, M., Peraire, J., and Patera, A., "A Posteriori Finite Element Bounds for Linear-Functional Outputs of Elliptic Partial Differential Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 150, 1997, pp. 289–312.

[23]Pierce, N.A., and Giles, M.B., "Adjoint Recovery of Superconvergent Functionals from PDE Approximations," *SIAM Review*, Vol. 42, No. 2, 2000, pp. 247–264.

[24]Venditti, D.A., and Darmofal, D.L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal of Computational Physics*, Vol. 164, 2000, pp. 204–227.

[25]Venditti, D.A., and Darmofal, D.L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, 2002, pp. 40–69.

[26]Muller, J.D., and Giles, M.B., "Solution Adaptive Mesh Refinement Using Adjoint Error Analysis," AIAA Paper 2001-2550, 2001.

[27]Venditti, D.A., "Grid Adaptation for Functional Outputs of Compressible Flow Simulations," Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, June 2002.

[28]Park, M.A., "Adjoint-Based, Three-Dimensional Error Prediction and Grid Adaptation," AIAA Paper 2002-3286, 2002.

[29]Venditti, D.A., and Darmofal, D.L., "Anisotropic Adaptation for Functional Outputs of Viscous Flow Simulations," Submitted to 16th AIAA Computational Fluid Dynamics Conference.

[30]Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O.C., "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, Vol. 72, 1987, pp. 249–266.

[31]Pirzadeh, S.Z., "A Solution-Adaptive Unstructured Grid Method by Grid Subdivision and Local Remeshing," *Journal of Aircraft*, Vol. 37, No. 5, September-October 2000, pp. 818–824.

[32]Park, M.T., and Kwon, O.J., "Unsteady Flow Computations Using a 3-D Parallel Unstructured Dynamic Mesh Adaptation Algorithm," AIAA Paper 2001-0865, 2001.

[33]Warren, G.P., Anderson, W.K., Thomas, J.L., and Krist, S.L., "Grid Convergence for Adaptive Methods," AIAA Paper 91-1592, 1991.

[34]Elliott, J., "Discrete Adjoint Analysis and Optimization with Overset Grid Modelling of the Compressible High-Re Navier-Stokes Equations," 6th Overset Grid and Solution Technology Symposium, Fort Walton Beach, FL, Oct. 2002.

[35]Anderson, W.K., and Bonhaus, D.L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No.1, 1994, pp. 1–21.

[36]Anderson, W.K., Rausch, R.D., and Bonhaus, D.L., "Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 128, 1996, pp. 391–408.

[37]Saad, Y., and Schultz, M.H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," SIAM Journal of Scientific and Statistical Computing, Vol. 7, No. 3, 1986, pp. 856–869.

[38]Spalart, P.R., and Allmaras, S.R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0429, 1992.

[39]Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[40]Van Leer, B., "Flux Vector Splitting for the Euler Equations," Lecture Notes in Physics, Vol. 170, 1982, pp. 501–512.

[41]Allmaras, S.R., "Multigrid for the 2-D Compressible Navier-Stokes Equations," AIAA Paper 99-3336, 1999.

[42]Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA-M6 Wing at Transonic Mach Numbers," Experimental Database for Computer Program Assessment, AGARD-AR-138, May 1979, pp. B1-1–B1-44.

[43]Trottenberg, U., Oosterlee, C., and Schuller, A., *Multigrid*, Academic Press, San Diego, 2001, p. 134.

[44]Mavriplis, D., "Directional Agglomeration Multigrid Techniques for High-Reynolds Number Viscous Flows," ICASE Report No. 98-7, January 1998.

[45]Redeker, G., "DLR-F4 Wing Body Configuration," AGARD-AR-303, Vol. 2, August 1994.

[46]Karypis, G., and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal of Scientific Computing*, Vol. 20, No. 1, 1998, pp. 359–392.

[47]Karypis, G., and Kumar, V., "Multilevel Algorithms for Multi-Constraint Graph Partitioning," University of Minnesota Technical Report No. 98-019, 1998.

[48]Levy, D.W., Zickuhr, T., Vassberg, J., Agrawal, S., Wahls, R.A., Pirzadeh, S., and Hemsch, M.J., "Summary of Data from the First AIAA CFD Drag Prediction Workshop," AIAA Paper 2002-0841, 2002.

[49]Kleb, W.L., Nielsen, E.J., and Gnoffo, P.A., "Collaborative Software Development in Support of Fast Adaptive AeroSpace Tools (FAAST)," Submitted to 16th AIAA Computational Fluid Dynamics Conference.

[50]Samareh, J.A., "A Novel Shape Parameterization Approach," NASA TM-1999-209116, May 1999.

[51]Pirzadeh, S., "Three-Dimensional Unstructured Viscous Grids by the

Advancing-Layers Method," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 43–49.

[52]Spaid, F.W., "High Reynolds Number, Multielement Airfoil Flowfield Measurements," *Journal of Aircraft*, Vol. 37, No. 3, 2000, pp. 499–507.

[53]Rumsey, C.L., Lee-Rausch, E.M., and Watson, R.D., "Three-Dimensional Effects on Multi-Element High Lift Computations," AIAA Paper 2002-0845, 2002.

[54]Alexandrov, N., Alter, S., Atkins, H., Bey, K., Bibb, K., Biedron, R., Carpenter, M., Cheatwood, F., Drummond, P., Gnoffo, P., Jones, W., Kleb, W., Lee-Rausch, E., Merski, R., Mineck, R., Nielsen, E., Park, M., Pirzadeh, S., Roberts, T., Samareh, J., Swanson, C., Thomas, J., Vatsa, V., Weilmuenster, J., White, J., Wood, W., and Yip, L., "Opportunities for Breakthroughs in Large-Scale Computational Simulation and Design," NASA TM-2002-211747, April 2002.